GARDA: a Diagnostic ATPG for Large Synchronous Sequential Circuits

F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy

Abstract*

The paper deals with automated generation of diagnostic test sequences for synchronous sequential circuits. An algorithm is proposed, named GARDA, which is suitable to produce good results with acceptable CPU time and memory requirements even for the largest benchmark circuits. The algorithm is based on Genetic Algorithms, and experimental results are provided which demonstrate the effectiveness of the approach.

1. Introduction

Diagnosis is the process of locating the fault responsible for a given faulty behavior: a popular method for the diagnosis of digital circuits lies in applying a Test Set to the faulty circuit, observing the output response, and then comparing them with the ones stored in the *fault dictionary* [ABFr90]. The success of such an approach mainly depends on the diagnostic capabilities of the Test Set, and some work has been done to devise viable techniques for the automatic generation of suitable Test Sets.

Given a sequential circuit, a Test Sequence T and two faults f_1 and f_2, T *distinguishes* f_1 and f_2 iff at least one input vector in T produces different output values in the faulty circuits for f_1 and f_2. All the faults which have not been distinguished by T belong to a same *Indistinguishability Class* [RFPa92].

Diagnostic Test Pattern Generation aims at producing a Test Set such that any couple of non-equivalent faults is distinguished by at least one Test Sequence belonging to the Test Set. The Indistinguishability Classes produced by such a Test Set coincide with the Fault Equivalence Classes [ABFr90] for the same circuit.

* This work has been partially supported by ESPRIT BRA 6575 *ATSEC*, and by the MURST 40% project *Affidabilità e Diagnostica in Elettronica*. Contact address: Paolo Prinetto, Politecnico di Torino, Dipartimento di Automatica e Informatica, Corso Duca degli Abruzzi 24, I-10129 Torino (Italy), e-mail `Paolo.Prinetto@polito.it`

Diagnostic Test Pattern Generation is even heavier than detection-oriented Test Pattern Generation, and Diagnostic ATPGs [CMPS90] [GMKo91] [CCCP92] produce Test Sets which partition the Fault List into Indistinguishability Classes. Each such class may include more Fault Equivalence Classes: in fact, faults in a same class have not yet been distinguished one from the other by any Test Sequence belonging to the Set, but a distinguishing Test Sequence may exist.

In this paper we propose GARDA (Genetic Algorithm for Diagnostic ATPG), a new algorithm for Diagnostic Test Pattern Generation. The algorithm is based on Genetic Algorithms (GAs) [Holl75], which have already been recognized to be effective for the generation of detection-oriented test sets [RHSP94] [RPGN94] [PRSR94]. Their main characteristics are:

- they have reduced memory requirements
- they allow the user to easily trade-off CPU time requirements and results accuracy
- they can exploit heuristics already developed for other approaches.

With respect to [PRSR94], several important changes have been introduced to take into account the different goal (diagnosys instead of detection): the main ones are the use of an *ad hoc* developed fault simulator, and the adoption of new evaluation and fitness functions.

Section 2 describes the diagnostic algorithm; Section 3 reports some experimental results and Section 4 draws some conclusions.

2. The Diagnostic ATPG Algorithm

The goal of GARDA is to produce a Test Set which partitions the Fault List into the highest number of Indistinguishability Classes. At the beginning, all the faults are grouped in a single class. Each time the algorithm generates a Test Sequence which distinguishes at least one couple of faults belonging to the same class, the Test Sequence is added to the Test Set, and the corresponding class is split.

The algorithm consists of three phases:

- *phase 1*: selection of a target class from the current Indistinguishability Classes;
- *phase 2*: generation of a sequence, if any, able to split the target class into two or more classes;
- *phase 3*: diagnostic fault simulation of the sequence and search for additional classes to be split.

The three phases are repeated until a pre-defined maximum number of iterations MAX_CYCLES has been reached. They will be analyzed in details in the next paragraphs.

2.1. The Genetic Algorithm for ATPG

Generation of a diagnostic sequence able to split a Indistinguishability Class is a search in the space of all the possible sequences applicable to the Primary Inputs (PIs) of the circuit. Applying GAs requires that both a suitable encoding for the generic solution, and an effective evaluation function be found.

As far as *encoding* is considered, GARDA assumes that an *individual* correspond to a *sequence* composed of a variable number of input *vectors* applied from the reset state. A *population* is a set of individuals.

Finding an effective *evaluation function* is a much more complex task. GARDA uses two heuristic parameters:

- the weighted number of gates with different values in the faulty circuits corresponding to the faults of a given class. The weight measures the observability of the gate it is associated with;
- the weighted number of Flip-Flops (whose inputs will hereinafter be referred to as *Pseudo Primary Outputs* or PPOs) with different values in the faulty circuits corresponding to the faults of a given class. The weight measures the observability of the Flip-Flop it is associated with.

The following function has been defined to rank sequences according to their distance from being diagnostic sequences with respect to a class; the function $h(v^j_k, s_j)$ estimates how close the class c_i is to being split by the k-th input vector v^j_k of the sequence s_j:

$$h(v^j_k, c_i) = k_1 \sum_{p=1}^{n_{gate}} w_p' d_p'(v^j_k, c_i) + k_2 \sum_{m=1}^{n_{FF}} w_m'' d_m''(v^j_k, c_i)$$

where:

- n_{gate} and n_{FF} are the number of gates and Flip-Flops, respectively
- w'_p and w''_m are the weights of the p-th gate and m-th Flip-Flop, respectively, and will be defined in the following
- the function $d'_p(v^j_k, c_i)$ returns 1 (0) iff, as a consequence of the application of v^j_k, two faults belonging to the class c_i exist, such that the value of the p-th gate is different (equal) in the two faulty circuits
- the function $d''_m(v^j_k, c_i)$ returns 1 (0) iff, as a consequence of the application of v^j_k, two faults belonging to the class c_i exist, such that the value of the m-th Flip-Flop is different (equal) in the two faulty circuits.

The values for k_1 and k_2 are experimentally found; in general, $k_2 > k_1$, as differences on Flip-Flops are normally more desirable than those on gates.

An evaluation function $H(s_j, c_i)$ is associated with each sequence s_j and class c_i; H corresponds to the maximum value of the function h defined above:

$$H(s_j, c_i) = \max_k(h(v^j_k, c_i)) \qquad v^j_k \in s_j$$

Once the individual encoding and the evaluation function have been defined, the whole process can be organized in two steps: first, several random sequences are generated until the evaluation function of a particular class becomes greater than a given threshold THRESH. Second, this class is chosen as the *target* one, and a diagnostic sequence for splitting it is generated. These two steps constitute phase 1 and phase 2, and require more detailed investigation.

2.1.1. Phase 1: choosing the target class

Sequences are randomly generated in groups of NUM_SEQ, and are all composed of L vectors. The diagnostic fault simulation of each sequence with respect to all the Indistinguishability Classes is performed and the evaluation function is computed. If no class produces a H greater than THRESH, a new set of NUM_SEQ random sequences is generated, whose length L is increased. Otherwise, the class with the maximum value of the evaluation function is selected as target class. The whole diagnostic ATPG process is stopped when a maximum number MAX_ITER of iterations is reached. Some classes can be split during this phase, and the corresponding sequences are inserted in the final set of test sequences.

It must be noted that phase 1 is purely random and does not exploit GAs, because neither the cross-over, nor the mutation operators are applied.

L is assigned an initial value L_{in} whose value is based on the topological characteristics of the circuit. L

is then increased in phase 1 as explained above, and is updated before any activation of phase 1 by using the length of the diagnostic sequence generated by the last phase 2.

2.1.2. Phase 2: Generating a Diagnostic Sequence for the Target Class

Phase 2 is based on a GA. Each sequence is an individual and NUM_SEQ sequences constitute a population. The initial population is composed of the last NUM_SEQ sequences generated in phase 1. The target class c_t, only, is considered in this phase.

The fitness function $F(s_j)$ is obtained from the evaluation function $H(s_j, c_t)$ via *linearization*: the individuals are sorted in decreasing order with respect to H, and the fitness value NUM_SEQ is assigned to the first individual, the value NUM_SEQ-1 to the second, and so on.

A new population is generated from the previous one through *evolution:* NEW_IND newly created individuals replace the worst individuals in the previous generation. The survival of the best NUM_SEQ-NEW_IND individuals from one generation to the next is thus ensured.

Evolution proceeds through two operators:

- the *cross-over* operator selects two parent individuals from the current population, randomly generates two numbers x_1 and x_2, and builds a new individual composed of the first x_1 vectors of the first parent and the last x_2 vectors of the second;
- the *mutation* operator acts on the newly generated test sequences with probability p_m and changes a single vector within it.

Candidates for the cross-over operator are selected on a probabilistic basis: the likelihood that an individual will be selected is proportional to its fitness, so that better sequences are more likely to provide vectors for the new individuals.

Once a new population has been generated, the fitness function is evaluated for the target class and for each sequence. The process is repeated until one of the following conditions is met:

- the target class is split: the corresponding sequence is then inserted in the final set of test sequences;
- a given maximum number of generations MAX_GEN is generated without splitting the class: such a class is then marked as *aborted* and its threshold THRESH is increased by a constant HANDICAP.

2.2. Diagnostic Fault Simulation

The computation of the evaluation function in phase 1 and 2, as well as the diagnostic fault simulation of the generated Test Sequence in phase 3, require an efficient diagnostic fault simulator. We developed an *ad hoc* tool which is based on the HOPE algorithm [LeHa92]. The following changes have been introduced to cope with the diagnostic purposes:

- all the PO values are computed for every simulated fault and every input vector
- a fault is dropped only when it has been distinguished from any other fault
- at the end of the simulation of each input vector, the PO values of faults belonging to the same class are compared, in order to check wheter the class can be split
- an additional data structure, which is dynamically updated during the ATPG process, is used to record fault partitioning in classes.

3. Experimental Results

GARDA has been implemented in ANSI-C and counts-up to about 4,000 lines of code. The large circuits in the ISCAS'89 standard set [BBKo89] have been considered.

Tab. 1 shows the results obtained running the tool on a SUN SPARCstation 2 with a 32 Mbyte memory. Only the largest ISCAS'89 circuits were considered. To the best of our knowledge, no previously proposed method is able to produce any diagnostic sequence for such circuits.

To evaluate how good the results are, we can proceed in two directions:

- for the smallest circuits [CCCP92] provides the exact number N_{FEC} of Fault Equivalence Classes; in this cases one can compare the number of classes we obtained with N_{FEC}. Tab. 2 shows that GARDA produces results not far from the exact ones.
- when N_{FEC} is not available, a comparison could be made with [RFPa92] for the Test Sets generated by two detection-oriented ATPGs, i.e., STG3 and HITEC. Unfortunately, [RFPa92] adopts a notion of distinguished faults based on a 3-valued logic, while GARDA uses the 0 and 1 values, only. However, the evaluation procedure are quite similar: we first group faults according

to the size of the Indistinguishability Class they belong to (Tab. 3). Column 2 contains the number of *Fully Distinguished Faults* (i.e., faults which have been distinguished from any other fault); column 7 the number of faults belonging to classes whose size is greater than 5. We then define the *k-Diagnostic Capability* (DC_k) of a Test Set as the percent of faults which belong to classes smaller than a given size k. The last column reports DC_6 , i.e., the percent number of faults belonging to classes smaller than 6. DC_6 corresponds to the percent number of faults for which a *reasonable* resolution capability is guaranteed. Today's technology prevents the exact computation of DC_6: no tool is in fact capable of partitioning the whole set of faults into exact Fault Equivalence Classes. However, the exact value of DC_6 for the largest circuits is expected to be generally low due to:

- the high (but unknown) number of untestable faults, which belong to a same Indistinguishability Class
- the large average size of classes.

Moreover, some circuits (like S9234 and S15850) are known to be critical for GA-based detection oriented ATPGs, too [PRSR94].

Memory occupation requirement is small, as it is substantially confined to storage of the sequences and to the space needed for the diagnostic fault simulation.

Effectiveness of the evolutionary approach is often evaluated by comparing its performance with that of a purely random one. In GARDA, phase 1 is random: the GA further increases the number of Indistinguishability Classes in phases 2 and 3. The percent ratio between the number of classes for which the last split occurred in phase 2 or 3, with respect to the total number of classes, varies from one circuit to another, but is greater than 60% for the largest circuits.

4. Conclusions

Generation of a Test Set with acceptable diagnostic capabilities is a challenging task when the largest circuits are considered, as the Test Set should distinguish any non-equivalent couple of faults. The task is thus much heavier than the one of generating detection-oriented test patterns. In this paper GARDA, a diagnostic ATPG for large synchronous sequential circuits, has been described. Our approach is based on Genetic Algorithms, and is able to produce good results with acceptable CPU time and memory results. A prototypical implementation has been developed, and for the first time, a Test Set with significant diagnostic capabilities has been generated for the largest benchmark circuits.

5. Acknowledgments

The authors wish to thank Giovanni Squillero for implementing GARDA.

6. References

[ABFr90] M. Abramovici, M. A. Breuer, A. D. Friedman, "*Digital systems testing and testable design*," Computer Science Press, New York, USA, 1990

[BBKo89] F. Brglez, D. Bryant, K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Proc. Int. Symp. on Circuits And Systems*, 1989, pp. 1929-1934

[CCCP92] G. Cabodi, P. Camurati, F. Corno, P. Prinetto, M. Sonza Reorda, "Sequential circuit diagnosis based on formal verification techniques," *Proc. International Test Conf., 1992*, pp. 187-196

[CMPS90] P. Camurati, D. Medina. P. Prinetto, M. Sonza Reorda, "A diagnostic test pattern generation algorithm," *Proc. Int. Test Conf., 1992*, pp. 52-58

[GMKo91] T. Grüning, U. Mahlstedt, H. Koopmeiners, "DIATEST: a fast diagnostic test pattern generator for combinational circuits," *Proc. Int. Conf. on Comp. Aided Design*, 1991, pp. 194-197

[Holl75] J.H. Holland, "Adaption in Natural and Artificial Systems," University of Michigan Press, Ann Arbor (USA), 1975

[LeHa92] H.K. Lee, D.S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits," *Proc. 29th Design Automation Conf.*, 1992, pp. 336-340

[PRSR94] P. Prinetto, M. Rebaudengo, M. Sonza Reorda, "An Automatic Test Pattern Generator for Large Sequential Circuits based on Genetic Algorithms,", *Proc. Int. Test Conf.*, 1994

[RFPa92] E.M. Rudnick, W.K. Fuchs, J.H. Patel, "Diagnostic Fault Simulation of Sequential Circuits," *Proc. Int. Test Conf.*, 1992, pp. 178-186

[RHSP94] E.M. Rudnick, J.G. Holm, D.G. Saab, J.H. Patel, "Application of Simple Genetic Algorithms to Sequential Circuit Test Generation," *Proc. European Design & Test Conf.*, 1994, pp. 40-45

[RPGN94] E.M. Rudnick, J.H. Patel, G.S. Greenstein, T.M. Niermann, "Sequential Circuit Test Generation in a Genetic Algorithm Framework," *Proc. Design Automation Conf.*, 1994, pp. 698-704

Circuit	# Indist. Classes	CPU time	# Sequences	# Vectors
S1196	1157	1.5h	228	7012
S1238	1172	1h	198	6319
S1423	489	3h	141	8101
S1488	973	1.5h	88	5645
S1494	938	1.5h	66	3764
S5378	2424	8h	158	4076
S9234	75	4h	9	50
S13207	749	10h	48	1360
S15850	296	10h	27	309
S35932	7347	14h	35	362
S38417	2251	16h	88	820
S38584	5250	18h	90	765

Tab. 1: Experimental results.

Circuit	# Indist. Classes	
	GARDA	*(CCCP92)*
S1196	1160	1200
S1238	1172	1223
S1488	973	1390
S1494	938	1396

Tab. 2: Comparison with the exact results.

Circuit	Number of Faults by Class Size						Tot.	DC_6
	1	*2*	*3*	*4*	*5*	*>5*	*[#]*	%
S1196	1107	96	6	4	5	24	1242	98.07
S1238	1098	132	6	20	0	99	1355	92.69
S1423	340	142	57	72	85	819	1515	45.94
S1488	846	190	63	28	10	349	1486	76.51
S1494	809	190	75	16	10	406	1506	73.04
S5378	1970	418	471	64	25	1655	4603	64.05
S9234	30	28	45	12	5	6807	6927	1.73
S13207	456	312	141	96	100	8710	9815	11.26
S15850	170	128	90	64	25	11242	11719	4.07
S35932	4350	3240	801	1452	75	29176	39094	25.37
S38417	1239	954	336	496	410	27745	31180	11.02
S38584	3603	1920	816	644	330	28993	36306	20.14

Tab. 3: Faults by Class size.